Distributed ETALIS Manual

Jia Ding

23.03.2013

Contents

1 About

Distributed-ETALIS is a distriuted event processing engine. It is based on **Storm** [1] (a distributed realtime computation system) and **Kafka** [2] (a distributed publish-subscribe messaging system). To set up the whole environment please follow the instruction in the next section.

2 Setting and Build

Storm cluster mode can only work under Linux, all the developments of distributed-ETALIS are under **Ubuntu** 12.04-LTS [3].

2.1 Setting up a Storm cluster

You can read the wiki page of storm (`https://github.com/nathanmarz/storm/wiki/Setting-up-a-Storm-cluster`) or read the following subsections.

2.1.1 Set up a Zookeeper cluster

Storm uses Zookeeper for coordinating the cluster. Zookeeper is not used for message passing, so the load Storm places on Zookeeper is quite low. Single node Zookeeper clusters should be sufficient for most cases, but if you want failover or are deploying large Storm clusters you may want larger Zookeeper clusters. Instructions for deploying Zookeeper are at (`http://zookeeper.apache.org/doc/r3.3.3/zookeeperAdmin.html`).

A few notes about Zookeeper deployment:

- It's critical that you run Zookeeper under supervision, since Zookeeper is fail-fast and will exit the process if it encounters any error case.
- It's critical that you set up a cron to compact Zookeeper's data and transaction logs. The Zookeeper daemon does not do this on its own, and if you don't set up a cron, Zookeeper will quickly run out of disk space.

2.1.2 Install dependencies on Nimbus and worker machines

Next you need to install Storm's dependencies on Nimbus and worker machines. These are:

- **ZeroMQ 2.1.7**
  Storm has been tested with ZeroMQ 2.1.7, and is the recommended ZeroMQ release that you install. You can download a ZeroMQ release from `http://download.zeromq.org/`. Install ZeroMQ should look something like this:

```
wget http://download.zeromq.org/zeromq-2.1.7.tar.gz
tar -xzf zeromq-2.1.7.tar.gz
cd zeromq-2.1.7
./configure
make
sudo make install
```

- **JZMQ**
  JZMQ is the Java bindings for ZeroMQ. JZMQ doesn't have any releases, so there is risk of a regression if you always install from the master branch. To prevent a regression from happening, you should instead install from this fork which is tested to work with Storm. Installing JZMQ should look something like this:

```
git clone https://github.com/nathanmarz/jzmq.git
cd jzmq
./autogen.sh
./configure
make
sudo make install
```

  To get the JZMQ build to work, you may need to do one or all of the following:

  1) Install Java dev package
  2) Set JAVA_HOME environment variable appropriately
  3) Upgrade autoconf on your machine
  4) If you run any errors when running ./configure, this thread (`http://stackoverflow.com/questions/3522248/how-do-i-compile-jzmq-for-zeromq-on-osx`) may provide a solution.

5) If you get any errors when running $\boxed{\text{make}}$, this thread (`https://github.com/zeromq/jzmq/issues/114`) may provide a solution.

- **Java 6**
- **Python 2.6.6**

These are the versions of the dependencies that have been tested with Storm. Storm may or may not work with different versions of Java and/or Python.

2.1.3 Download and extract a Storm release to Nimbus and worker machines

Next, download a Storm release and extract the zip file somewhere on Nimbus and each of the worker machines. The Storm releases can be downloaded from `https://github.com/nathanmarz/storm/downloads`.

2.1.4 Fill in mandatory configurations into storm.yaml file

The Storm release contains a file at $\boxed{\text{conf/storm.yaml}}$ that configures the Storm daemons. There's a few configurations that are mandatory to get a working cluster:

1) **storm.zookeeper.servers**: This is a list of the hosts in the Zookeeper cluster for your Storm cluster. It should look something like:

```
storm.zookeeper.servers:
  - "111.222.333.444"
  - "555.666.777.888"
```

   If the port that your Zookeeper cluster uses is different than the default, you should set **storm.zookeeper.port** as well.

2) **storm.local.dir**: The Nimbus and Supervisor daemons require a directory on the local disk to store small amounts of state (like jars, confs, and things like that). You should create that directory on each machine, give it proper permissions, and then fill in the directory location using this config. For example:

```
storm.local.dir: "/mnt/storm"
```

3) **java.library.path**: This is the load path for the native libraries that Storm uses (ZeroMQ and JZMQ). The default of ”/usr/local/lib:/opt/local/lib:/usr/lib” should be fine for most installations, so you probably don't need to set this config.

4) **nimbus.host**: The worker nodes need to know which machine is the master in order to download topology jars and confs. For example:

```
nimbus.host: ”111.222.333.44”
```

5) **supervisor.slots.ports**: For each worker machine, you configure how many workers run on that machine with this config. Each worker uses a single port for receiving messages, and this setting defines which ports are open for use. If you define five ports here, then Storm will allocate up to five workers to run on this machine. If you define three ports, Storm will only run up to three. By default, this setting is configured to run 4 workers on the ports 6700, 6701, 6702, and 6703. For example:

```
supervisor.slots.ports:
    - 6700
    - 6701
    - 6702
    - 6703
```

2.1.5 Launch daemons under supervision using storm script and a supervisor of your choice

The last step is to launch all the Storm daemons. It is critical that you run each of these daemons under supervision. Storm is a fail-fast system which means the processes will halt whenever an unexpected error is encountered. Storm is designed so that it can safely halt at any point and recover correctly when the process is restarted. This is why Storm keeps no state in-process – if Nimbus or the Supervisors restart, the running topologies are unaffected. Here's how to run the Storm daemons:

- **Nimbus**: Run the command ”bin/storm nimbus” under supervision on the master machine.

- **Supervisor**: Run the command "bin/storm supervisor" under supervision on each worker machine. The supervisor daemon is responsible for starting and stopping worker processes on that machine.
- **UI**: Run the Storm UI (a site you can access from the browser that gives diagnostics on the cluster and topologies) by running the command "bin/storm ui" under supervision. The UI can be accessed by navigating your web browser to http://nimbus host:8080.

As you can see, running the daemons is very straightforward. The daemons will log to the logs/ directory in wherever you extracted the Storm release.

2.2 Setting up Kafka

You can download a release of Kafka from `http://kafka.apache.org/downloads.html` and unpack it on nimbus and each worker machines. The current stable version is *0.7.2-incubating.*

2.3 Install SWI-Prolog

You should follow this link (`http://www.swi-prolog.org/build/Debian.html`) to install **SWI-Prolog**. When you install it, you should remember to enable the *jpl* and *shared* flag. After you install the SWI-Prolog, you should specify the path for the **JPL** library. For example:

```
export LD_LIBRARY_PATH=/usr/local/lib/swipl-6.3.10/lib/x86_64-linux
```

2.4 Setting up jtalis-distribute

Jtalis-distribute is a distributed version of Jtalis, which is a Java wrapper for **ETALIS** [5] engine. To set up the jtalis-distribute you need to install **Subversion** and **Maven**.
You should checkout the source from `http://etalis.googlecode.com/svn/dEtalis/jtalis-distribute` and build it:

```
svn chekcout http://etalis.googlecode.com/svn/dEtalis/jtalis-distribute
cd jtalis-distribute/scripts
```

```
./build.sh
```

2.5 Setting up distributed-Etalis

You should checkout the source from `http://etalis.googlecode.com/svn/dEtalis/jtalis-storm-zstream`. Then use **Maven** to build it.

```
svn chekcout http://etalis.googlecode.com/svn/dEtalis/jtalis-storm-zstream
cd jtalis-storm-zstream
mvn install
```

In the target folder, you will find *jtalis-storm-zstream-0.0.1-SNAPSHOT.jar*.

2.6 Run the Demo

Follow the below instructions to run the demo:

1) **Start Zookeeper Server, Storm Nimbus and Storm UI**

   i) Login the Storm master node

   ```
   ssh -i etalis01.pem ubuntu@141.52.160.167
   ```

   ii) Start Zookeeper Server

   ```
   cd ~/storm/kafka-0.7.2-incubating-src/
   sudo ./bin/zookeeper-server-start.sh config/zookeeper.properties
   ```

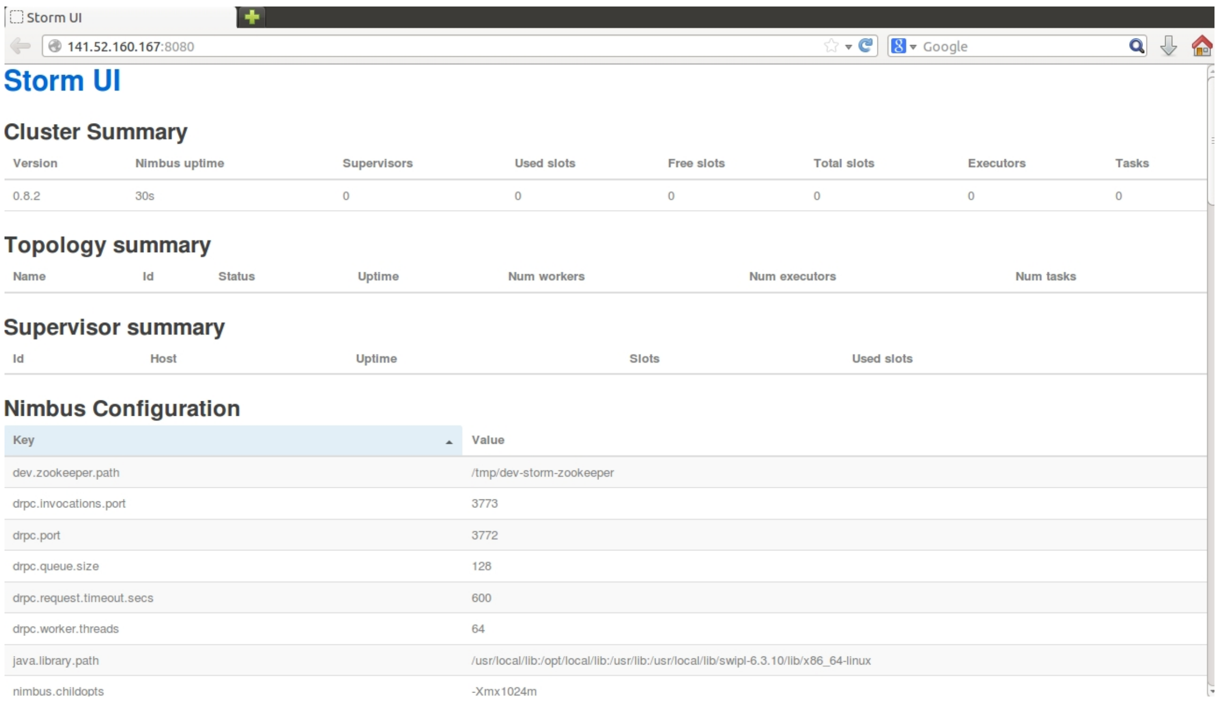
   iii) Start Storm Nimbus

   ```
   cd ~/storm/storm-0.8.2/
   sudo ./bin/storm nimbus
   ```

iv) Start Storm UI

```
cd ~/storm/storm-0.8.2/
sudo ./bin/storm ui
```

After ui started, open a web browser and access the Storm UI with this address (see Fig. 1):
`141.52.160.167:8080`



Storm UI

Cluster Summary

Version	Nimbus uptime	Supervisors	Used slots	Free slots	Total slots	Executors	Tasks
0.8.2	30s	0	0	0	0	0	0

Topology summary

Name	Id	Status	Uptime	Num workers	Num executors	Num tasks

Supervisor summary

Id	Host	Uptime	Slots	Used slots

Nimbus Configuration

Key	Value
dev.zookeeper.path	/tmp/dev-storm-zookeeper
drpc.invocations.port	3773
drpc.port	3772
drpc.queue.size	128
drpc.request.timeout.secs	600
drpc.worker.threads	64
java.library.path	/usr/local/lib:/opt/local/lib:/usr/lib:/usr/local/lib/swipl-6.3.10/lib/x86_64-linux
nimbus.childopts	-Xmx1024m

Figure 1: Storm UI

2) **Start Storm Supervisor daemon**
Start Storm Supervisor daemon on each VM.

```
ssh -i etalis01.pem ubuntu@141.52.160.167
cd ~/storm/storm-0.8.2/
sudo ./bin/storm supervisor
```

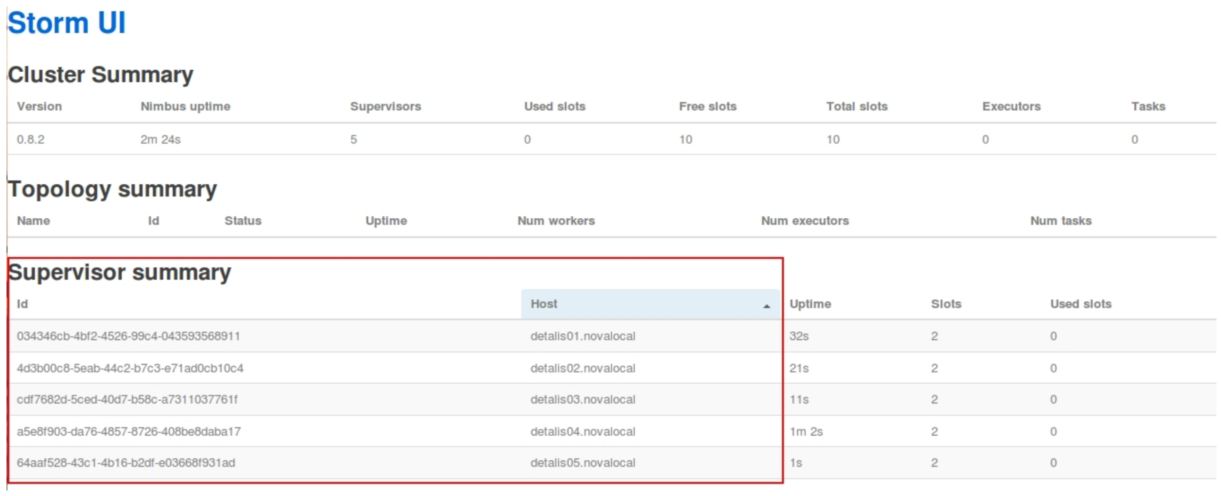
```
ssh -i etalis02.pem ubuntu@141.52.160.145
cd ~/storm/storm-0.8.2/
sudo ./bin/storm supervisor

ssh -i etalis03.pem ubuntu@141.52.160.146
cd ~/storm/storm-0.8.2/
sudo ./bin/storm supervisor

ssh -i etalis04.pem ubuntu@141.52.160.155
cd ~/storm/storm-0.8.2/
sudo ./bin/storm supervisor

ssh -i etalis05.pem ubuntu@141.52.160.156
cd ~/storm/storm-0.8.2/
sudo ./bin/storm supervisor
```

After start the 5 Supervisor on 5 VM, you can see the this on Storm UI (see Fig. 2).



Storm UI

Cluster Summary

Version	Nimbus uptime	Supervisors	Used slots	Free slots	Total slots	Executors	Tasks
0.8.2	2m 24s	5	0	10	10	0	0

Topology summary

Name	Id	Status	Uptime	Num workers	Num executors	Num tasks

Supervisor summary

Id	Host	Uptime	Slots	Used slots
034346cb-4bf2-4526-99c4-043593568911	detalis01.novalocal	32s	2	0
4d3b00c8-5eab-44c2-b7c3-e71ad0cb10c4	detalis02.novalocal	21s	2	0
cdf7682d-5ced-40d7-b58c-a7311037761f	detalis03.novalocal	11s	2	0
a5e8f903-da76-4857-8726-408be8daba17	detalis04.novalocal	1m 2s	2	0
64aaf528-43c1-4b16-b2df-e03668f931ad	detalis05.novalocal	1s	2	0

Figure 2: Storm Supervisor

3) Deploy the Demo Topology

i) Compile the source code to .jar file

```
ssh -i etalis01.pem ubuntu@141.52.160.167
cd dEtalis/
svn update
cd jtalis-storm-zstream/
mvn install
```

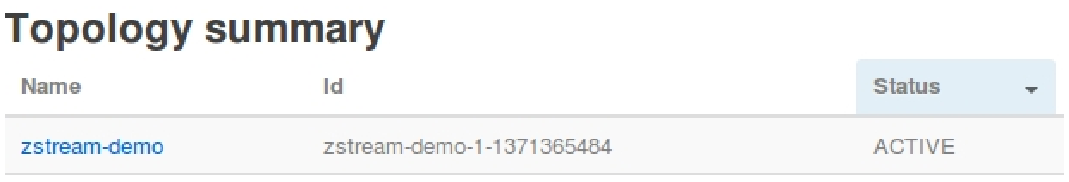
ii) Deploy the Topology to Storm

```
cd ~/storm/storm-0.8.2/
./bin/storm jar ~/dEtalis/jtalis-storm-zstream/target/
    jtalis-storm-zstream-0.0.1.jar com.jtalis.storm.
    zstream.topology.ZStreamDemoTopology 10 300 1000
```

ZStreamDemoTopology has 3 parameters:

* Number of workers: it depends on how many slots that storm has.
* Atom Events generate time interval: it sets every a period time to generate a atom event.
* Time Window: it sets the time window of the operator.

After you deploy the Topology, you can see it on the Storm UI (see Fig. 3).

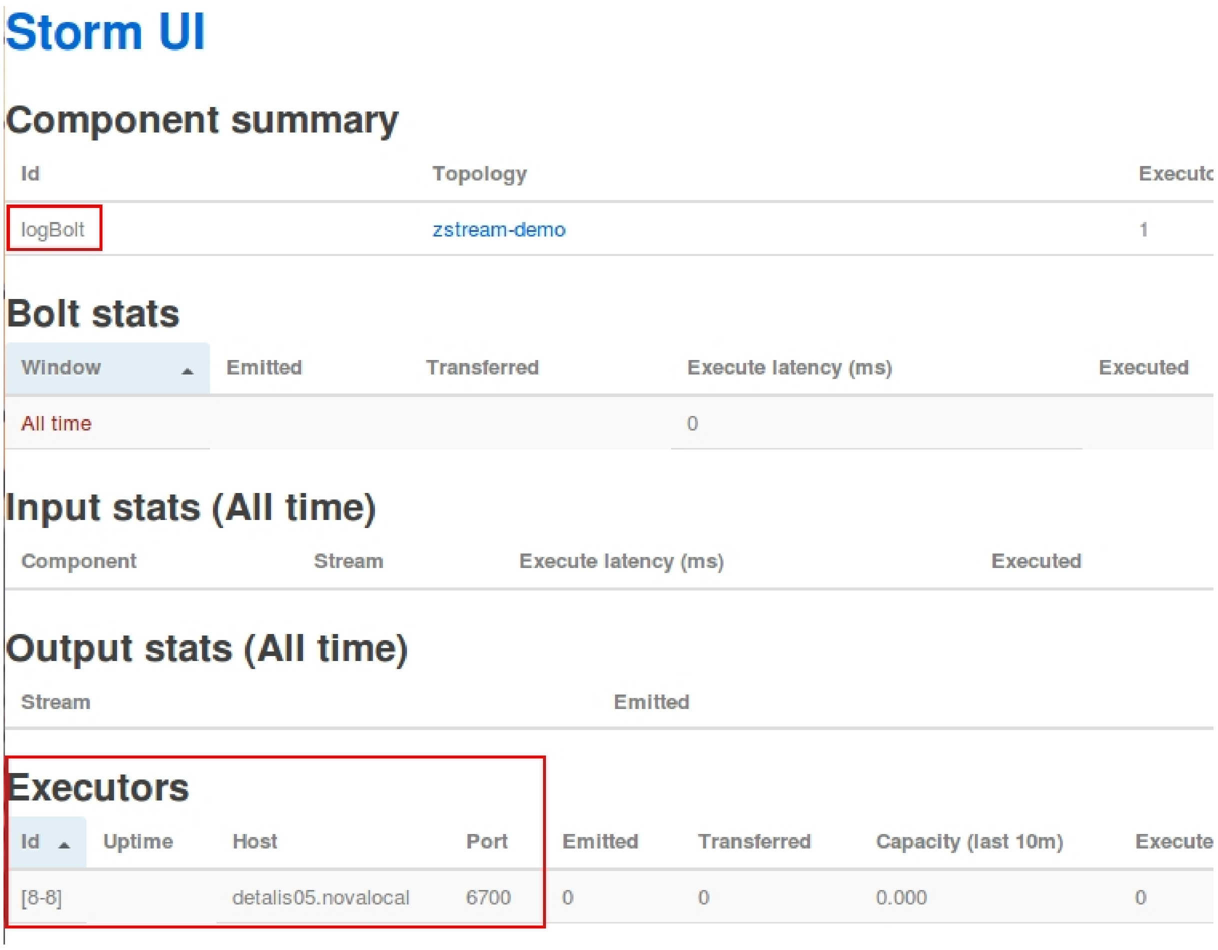


Topology summary

Name	Id	Status
zstream-demo	zstream-demo-1-1371365484	ACTIVE

Figure 3: Storm Topology

The result is stored by a component of this Topology, the name is $\boxed{\text{logBolt}}$. In Storm UI you can find, on which VM and on which Worker it works (see Fig. 4).
Here the logBolt is working on VM 5 and the worker id is 6700. You can login the VM 5 and to check the results (see Fig. 5).



Storm UI

Component summary

Id	Topology	Executc
logBolt	zstream-demo	1

Bolt stats

Window	Emitted	Transferred	Execute latency (ms)	Executed
All time			0	

Input stats (All time)

Component	Stream	Execute latency (ms)	Executed

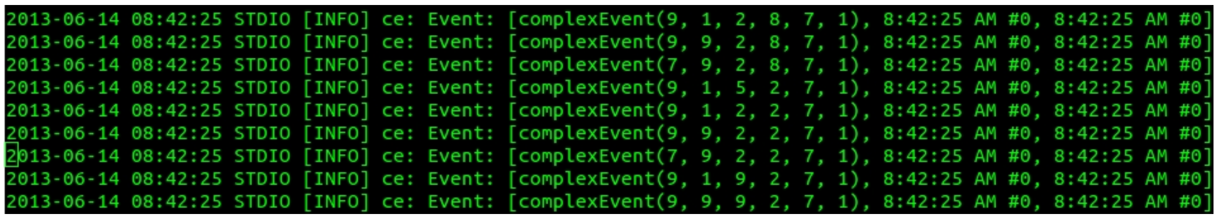
Output stats (All time)

Stream	Emitted

Executors

Id	Uptime	Host	Port	Emitted	Transferred	Capacity (last 10m)	Execute
[8-8]		detalis05.novalocal	6700	0	0	0.000	0

Figure 4: Storm logBolt



```
2013-06-14 08:42:25 STDIO [INFO] ce: Event: [complexEvent(9, 1, 2, 8, 7, 1), 8:42:25 AM #0, 8:42:25 AM #0]
2013-06-14 08:42:25 STDIO [INFO] ce: Event: [complexEvent(9, 9, 2, 8, 7, 1), 8:42:25 AM #0, 8:42:25 AM #0]
2013-06-14 08:42:25 STDIO [INFO] ce: Event: [complexEvent(7, 9, 2, 8, 7, 1), 8:42:25 AM #0, 8:42:25 AM #0]
2013-06-14 08:42:25 STDIO [INFO] ce: Event: [complexEvent(9, 1, 5, 2, 7, 1), 8:42:25 AM #0, 8:42:25 AM #0]
2013-06-14 08:42:25 STDIO [INFO] ce: Event: [complexEvent(9, 1, 2, 2, 7, 1), 8:42:25 AM #0, 8:42:25 AM #0]
2013-06-14 08:42:25 STDIO [INFO] ce: Event: [complexEvent(9, 9, 2, 2, 7, 1), 8:42:25 AM #0, 8:42:25 AM #0]
2013-06-14 08:42:25 STDIO [INFO] ce: Event: [complexEvent(7, 9, 2, 2, 7, 1), 8:42:25 AM #0, 8:42:25 AM #0]
2013-06-14 08:42:25 STDIO [INFO] ce: Event: [complexEvent(9, 1, 9, 2, 7, 1), 8:42:25 AM #0, 8:42:25 AM #0]
2013-06-14 08:42:25 STDIO [INFO] ce: Event: [complexEvent(9, 9, 9, 2, 7, 1), 8:42:25 AM #0, 8:42:25 AM #0]
```

Figure 5: Storm Log

```
ssh -i etalis05.pem ubuntu@141.52.160.156
cd ~/storm/storm-0.8.2/logs/
nano worker-6700.log
```

4) **Undeploy the Demo Topology**

```
./bin/storm kill zstream-demo
```

References

[1] http://storm-project.net/.

[2] http://kafka.apache.org/.

[3] http://www.ubuntu.com/.

[4] https://github.com/nathanmarz/storm/wiki/Setting-up-a-Storm-cluster.

[5] http://code.google.com/p/etalis/.